Amendments To The Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Original) A system for generating floating-point test-cases for verifying the operation of a floating-point arithmetic unit, the system comprising a processing unit which includes:

(a) an exponent generator, for generating floating-point exponents;

(b) a significand generator, for generating floating-point significands; and

(c) a fixed-point generator coupled to said exponent generator and to said signficand generator;

wherein said processing unit is configured to receive a specified arithmetic operation, a specified rounding mode, at least one input operand mask, and an output result mask; and wherein said processing unit is configured to output a set of floating-point numbers which includes at least one input operand compatible with said at least one input operand mask, and an output result compatible with said output result mask; and wherein said output result corresponds to said specified arithmetic operation on said at least one input operand for said specified rounding mode.

2. (Currently Amended) ~~A program of instructions in data storage executable by a machine for emulating the system of claim 1.~~ A data storage storing a program of instructions executable by a machine for emulating a system for generating floating-point test-cases for verifying the operation of a floating-point arithmetic unit, the system comprising a processing unit which includes:

(a) an exponent generator, for generating floating-point exponents;

(b) a significand generator, for generating floating-point significands; and

(c) a fixed-point generator coupled to said exponent generator and to said signficand generator;

wherein said processing unit is configured to receive a specified arithmetic operation, a specified rounding mode, at least one input operand mask, and an output result mask; and wherein said processing unit is configured to output a set of floating-point numbers which includes at least one input operand compatible with said at least one input operand mask, and an output result compatible with said output result mask; and wherein said output result corresponds to said specified arithmetic operation on said at least one input operand for said specified rounding mode.

3. (Original) A system for generating floating-point test-cases for verifying the operation of a floating-point arithmetic unit, the system comprising a processing unit which includes:

(a) an exponent generator, for generating floating-point exponents;

(b) a significand generator, for generating floating-point significands; and

(c) a fixed-point generator coupled to said exponent generator and to said signficand generator;

wherein said processing unit is configured to receive a specified arithmetic operation selected from a group that includes addition and subtraction, a specified rounding mode, a first input operand mask, a second input operand mask, and an output result mask; and wherein said processing unit is configured to output a set of floating-point numbers which includes a first input operand compatible with said first input operand mask, a second input operand compatible with said second input operand mask, and an output result compatible with said output result mask; and wherein said output result corresponds to said specified arithmetic operation on said first input operand and said second input operand for said specified rounding mode.

4. (Currently Amended) ~~A program of instructions in data storage executable by a machine for emulating the system of claim 3.~~ A data storage storing a program of instructions executable by a machine for emulating a system for generating floating-point test-cases for verifying the operation of a floating-point arithmetic unit, the system comprising a processing unit which includes:

(a) an exponent generator, for generating floating-point exponents;

(b) a significand generator, for generating floating-point significands; and

(c) a fixed-point generator coupled to said exponent generator and to said signficand generator;

wherein said processing unit is configured to receive a specified arithmetic operation selected from a group that includes addition and subtraction, a specified rounding mode, a first input operand mask, a second input operand mask, and an output result mask; and wherein said processing unit is configured to output a set of floating-point numbers which includes a first input operand compatible with said first input operand mask, a second input operand compatible with said second input operand mask, and an output result compatible with said output result mask; and wherein said output result corresponds to said specified arithmetic operation on said first input operand and said second input operand for said specified rounding mode.

5. (Original) The system of claim 3, wherein said fixed-point generator has two addends and a carry sequence representing the carries from the addition of successive digits of said addends, wherein said carry sequence is compatible with a carry sequence mask.

6. (Original) The system of claim 3, said significand generator further comprising:

(d) an addition significand generator, for generating floating-point significands for said addition operation; and

(e) a subtraction significand generator, for generating floating-point significands for said subtraction operation.

7. (Original) The system of claim 3, wherein said first input operand has a first input operand exponent, said

second input operand has a second input operand exponent, and said output result has an output result exponent, said exponent generator further comprising:

(d) a definite exponent generator, for generating floating-point exponents wherein said output result exponent is a definite amount different from either of said first input operand exponent and said second input operand exponent; and

(e) an indefinite exponent generator, for generating floating-point exponents wherein said output result exponent is not a definite amount different from either of said first input operand exponent and said second input operand exponent.

8. (Original) The system of claim 3, wherein said exponent generator is a biased exponent generator, for generating biased floating-point exponents.

9. (Original) The system of claim 8, wherein said first input operand has a first input operand biased exponent, said second input operand has a second input operand biased exponent, and said output result has an output result biased exponent, said biased exponent generator further comprising:

(d) a definite biased exponent generator, for generating biased floating-point exponents wherein said output result biased exponent is a definite amount different from either of said first input operand biased exponent and said second input operand biased exponent and

(e) an indefinite biased exponent generator, for generating biased floating-point exponents wherein said output result biased exponent is not a definite amount different from either of said first input operand biased exponent and said second input operand biased exponent.

10. (Original) The system of claim 8, further comprising an unbiased exponent shift calculator for computing an unbiased exponent shift from a biased exponent shift.

11. (Original) A method of seeking a solution, if a solution exists, to a specified mathematical condition, wherein the solution is used in constructing a floating-point test-case for verifying the operation of a floating-point arithmetic unit, wherein a complete generated test case is a set of floating-point numbers for a specified arithmetic operation and a specified rounding mode, and wherein a generated test case includes at least one input operand and an output result; and wherein an input operand is compatible with an operand mask, and the output result is compatible with an output result mask; the method comprising the steps of:

(a) preparing a list of choices upon which the solution is based;

(b) testing whether said list of choices is empty;

(c) outputting, if said list of choices is empty, that no solution exists;

(d) randomly choosing, if said list of choices is not empty, a choice of said list as a selection;

(e) searching for a solution to the specified mathematical condition, based on said selection;

(f) outputting, if said searching was successful, said solution;

(g) erasing, if said searching was not successful, said selection from said list; and

(h) repeating step (a) through step (g) until outputting occurs.

12. (Currently Amended) ~~A program of instructions in data storage executable by a machine for performing the method of claim 11.~~ A data storage storing a program of instructions executable by a machine for performing a method of seeking a solution, if a solution exists, to a specified mathematical condition, wherein the solution is used in constructing a floating-point test-case for verifying the operation of a floating-point arithmetic unit, wherein a complete generated test case is a set of floating-point numbers for a specified arithmetic operation and a specified rounding mode, and

wherein a generated test case includes at least one input operand and an output result; and wherein an input operand is compatible with an operand mask, and the output result is compatible with an output result mask; the method comprising the steps of:

(d) preparing a list of choices upon which the solution is based;

(e) testing whether said list of choices is empty;

(f) outputting, if said list of choices is empty, that no solution exists;

(g) randomly choosing, if said list of choices is not empty, a choice of said list as a selection;

(h) searching for a solution to the specified mathematical condition, based on said selection;

(i) outputting, if said searching was successful, said solution;

(j) erasing, if said searching was not successful, said selection from said list; and

(k) repeating step (a) through step (g) until outputting occurs.

13. (Original) A method of seeking a solution, if a solution exists, to a specified mathematical condition, wherein the solution is used in constructing a floating-point test-case for verifying the operation of a floating-point arithmetic unit, wherein a complete generated test case is a set of floating-point numbers for a specified arithmetic operation selected from a group including addition and subtraction, and for a specified rounding mode, and wherein a generated test case includes a first input operand, a second input operand, and an output result; and wherein the first input operand is compatible with a first input operand mask, the second input operand is compatible with a second input operand mask, and the output result is compatible with an output result mask; the method comprising the steps of:

(a) preparing a list of choices upon which the solution is based;

(b) testing whether said list of choices is empty;

(c) outputting, if said list of choices is empty, that no solution exists;

(d) randomly choosing, if said list Df choices is not empty, a choice of sail list as a selection;

(e) searching for a solution to the specified mathematical condition, based on said selection;

(f) outputting, if said searching was successful, said solution;

(g) erasing, if said searching was not successful, said selection from said list; and

(h) repeating step (a) through step (g) until outputting occurs.

14. (Currently Amended) ~~A program of instructions in data storage executable by a machine for performing the method of claim 13.~~ A data storage storing a program of instructions executable by a machine for performing the method of seeking a solution, if a solution exists, to a specified mathematical condition, wherein the solution is used in constructing a floating-point test-case for verifying the operation of a floating-point arithmetic unit, wherein a complete generated test case is a set of floating-point numbers for a specified arithmetic operation selected from a group including addition and subtraction, and for a specified rounding mode, and wherein a generated test case includes a first input operand, a second input operand, and an output result; and wherein the first input operand is compatible with a first input operand mask, the second input operand is compatible with a second input operand mask, and the output

result is compatible with an output result mask; the method comprising the steps of:

(a) preparing a list of choices upon which the solution is based;

(b) testing whether said list of choices is empty;

(c) outputting, if said list of choices is empty, that no solution exists;

(d) randomly choosing, if said list of choices is not empty, a choice of said list as a selection;

(e) searching for a solution to the specified mathematical condition, based on said selection;

(f) outputting, if said searching was successful, said solution;

(g) erasing, if said searching was not successful, said selection from said list; and

(h) repeating step (a) through step (g) until outputting occurs.

15. (Original) The method of claim 13, wherein said list of choices contains an exponent shift.

16. (Original) The method of claim 13, wherein the solution is a set of floating-point numbers.

17. (Original) The method of claim 13, wherein the solution is an exponent.

18. (Original) The method of claim 13, wherein the solution is a significand.

19. (Original) The method of claim 18, wherein said list of choices contains a tails triplet.

20. (Original) A method of generating a set of fixed-point numbers containing a first addend, a second addend, and a sum, wherein the first addend is compatible with a first addend mask, the second addend is compatible with a second addend mask, the sum is compatible with a sum mask, and wherein the addition of the first addend and the second addend results in a carry sequence of carry bits, wherein each carry bit has a unique index in the carry sequence, wherein the carry sequence is compatible with a carry sequence mask and wherein each carry bit has a value in the group consisting of 0, 1, and 2, and wherein there exists a boundary index in the carry sequence corresponding to the lowest index of a carry bit having the value 2; the method comprising the steps of:

(a) constructing a list of possible boundary indices;

(b) testing whether said list is empty;

(c) outputting, if said list is empty, that no solution exists;

(d) randomly choosing, if said list is not empty, a boundary index from said list as a selection;

(e) searching for a carry sequence based on said selection, which is compatible with the carry sequence mask;

(f) erasing, if said searching was not successful, said selection from said list;

(g) constructing, if said searching was successful, a first addend compatible with tie first addend mask, a second addend compatible with the second addend mask, and a sum compatible with the sum mask;

(h) outputting said first addend, said second addend, said sum, and said carry sequence; and

(i) repeating step (a) through step (h) until outputting occurs.

21. (Currently Amended) ~~A program of instructions in data storage executable by a machine for performing the method of claim 20.~~ A data storage storing a program of instructions executable by a machine for performing the method of generating a set of fixed-point numbers containing a first addend, a second addend, and a sum, wherein the first addend is compatible with a first addend mask, the second addend is

compatible with a second addend mask, the sum is compatible with a sum mask, and wherein the addition of the first addend and the second addend results in a carry sequence of carry bits, wherein each carry bit has a unique index in the carry sequence, wherein the carry sequence is compatible with a carry sequence mask and wherein each carry bit has a value in the group consisting of 0, 1, and 2, and wherein there exists a boundary index in the carry sequence corresponding to the lowest index of a carry bit having the value 2; the method comprising the steps of:

(a) constructing a list of possible boundary indices;

(b) testing whether said list is empty;

(c) outputting, if said list is empty, that no solution exists;

(d) randomly choosing, if said list is not empty, a boundary index from said list as a selection;

(e) searching for a carry sequence based on said selection, which is compatible with the carry sequence mask;

(f) erasing, if said searching was not successful, said selection from said list;

(g) constructing, if said searching was successful, a first addend compatible with the first addend mask, a second addend compatible with the second addend mask, and a sum compatible with the sum mask;

(h) outputting said first addend, said second addend, said sum, and said carry sequence; and

(i) repeating step (a) through step (h) until outputting occurs.